

FIG. 1

3/14

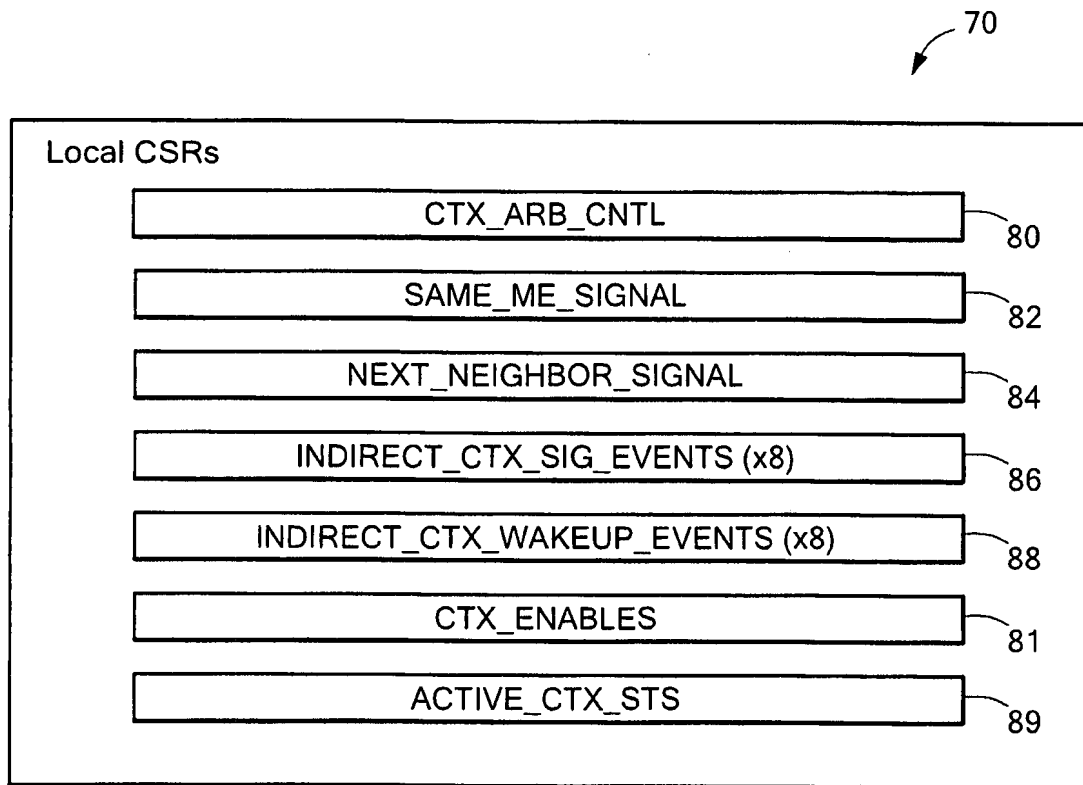
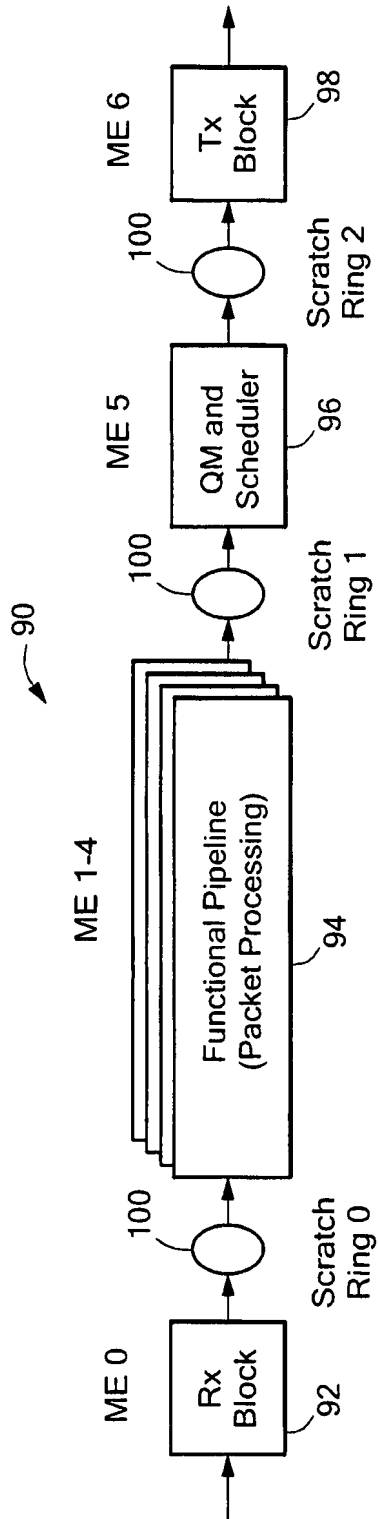
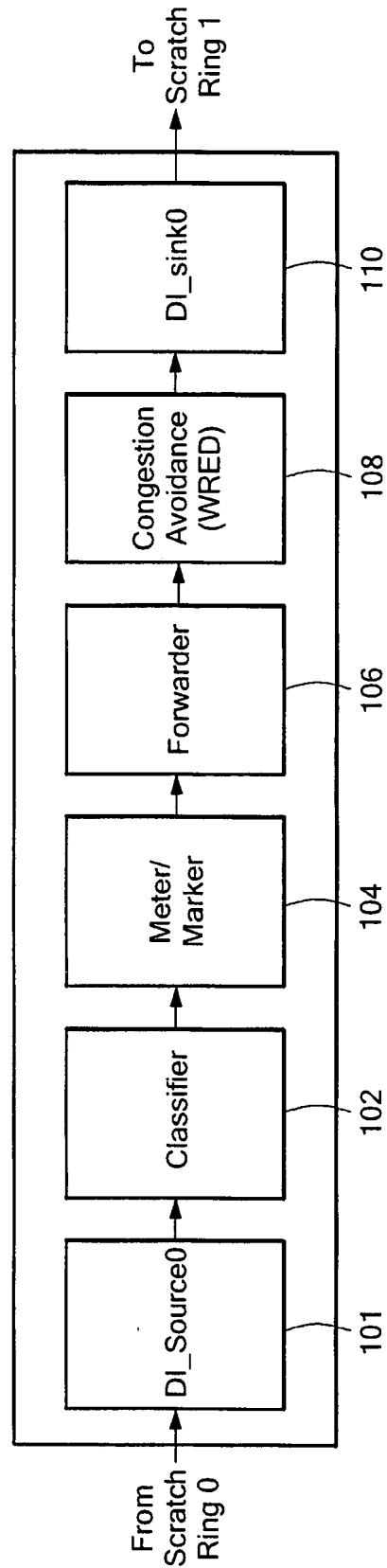


FIG. 3

4/14

**FIG. 4**

94

**FIG. 5**

120

Register Name	Latency in terms of instructions			Usage	Usage Latency Comments
	Write	Read			
SAME_ME_SIGNAL 82	3	2		8	The same ME will be signaled 8 cycles after the CSR write.

122 124 126

FIG. 6

6/14

```

wred 0
{
    if(ctx0=0)
    {
        //Wait for signal from previous ME and thread 7

        wait_for_all (&next_thread_signal, &wred_next_me_sig);
        cam_clear 0;
    }
    else
    {
        wait_for_all (&next_thread_signal);
    }
    .....
    .....
    //WRED packet processing
    .....
    signal_next_thread 0 //Instruction 1
    .....

    //There is a minimum of 3 cycles delay between instruction 1 and instruction 2
    //to allow the signal to propagate and to ensure thread execution sequence.
    .....
    //Wait for previous thread signal
    wait_for_sig (&sig); //Instruction 2
    .....
    if(ctx 0=7)
    {
        //Signal next ME

        cap_fast_write (wred_me_sig_csr, csr_interthread_sig);
    }
    else
    {
        //The thread give up the context voluntarily at this point to ensure that
        //thread 7 gets control as early as possible. If no context swap occurs
        //here the thread would continue to execute non-critical section code or
        //    next microblock, thereby delaying thread 7 getting the control.
        ctx_arb(voluntary);
    }

    //Critical section processing ends

    //Non-critical section code or code for next microblock begins

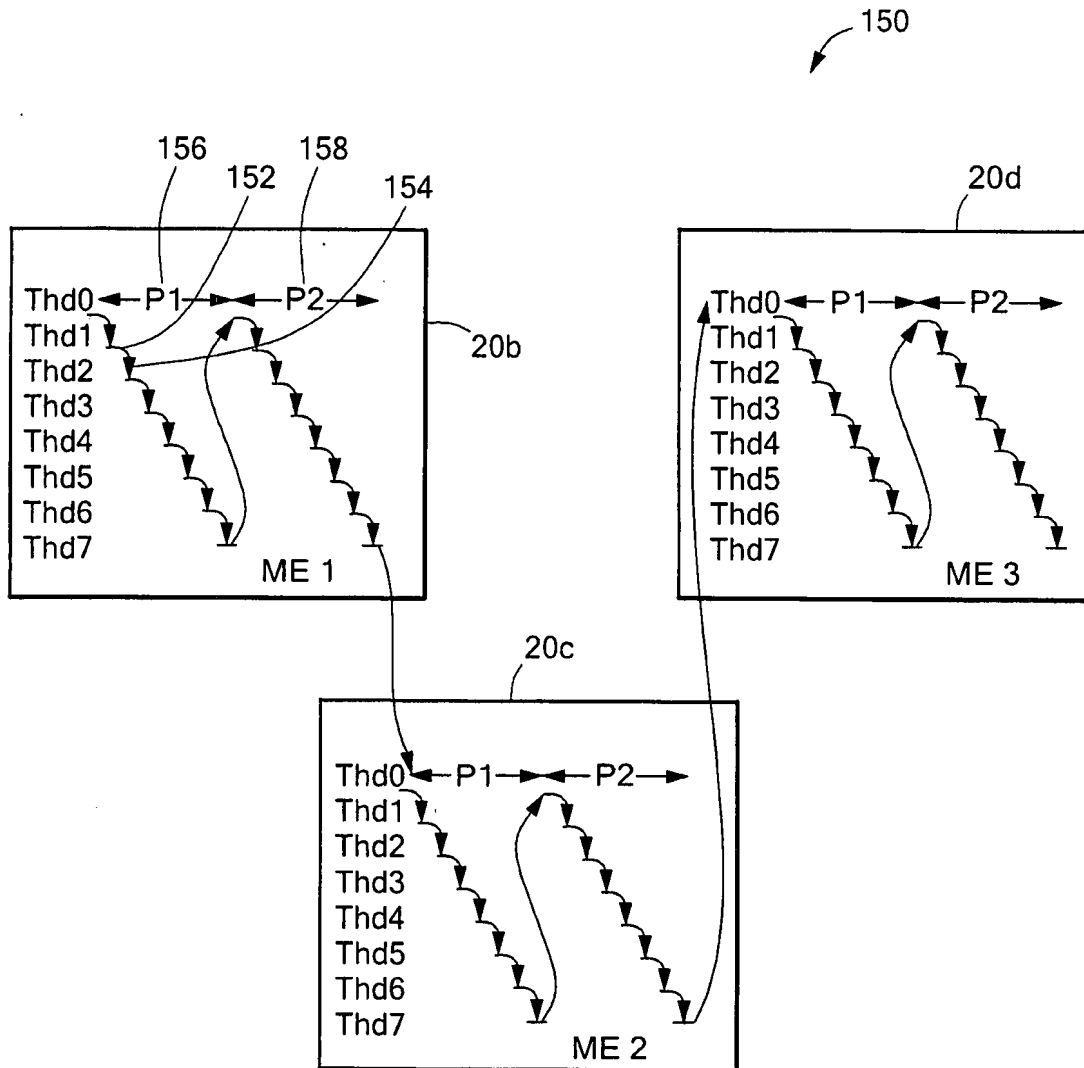
```

Diagram annotations:

- 130: Arrow pointing to the initial code block.
- 132: Bracket grouping the `wait_for_all` and `cam_clear` statements.
- 134: Arrow pointing to `signal_next_thread 0`.
- 136: Arrow pointing to `wait_for_sig (&sig);`.
- 138: Bracket grouping the `cap_fast_write` statement.
- 140: Arrow pointing to `ctx_arb(voluntary);`.

FIG. 7

7/14

**FIG. 8**

14/14

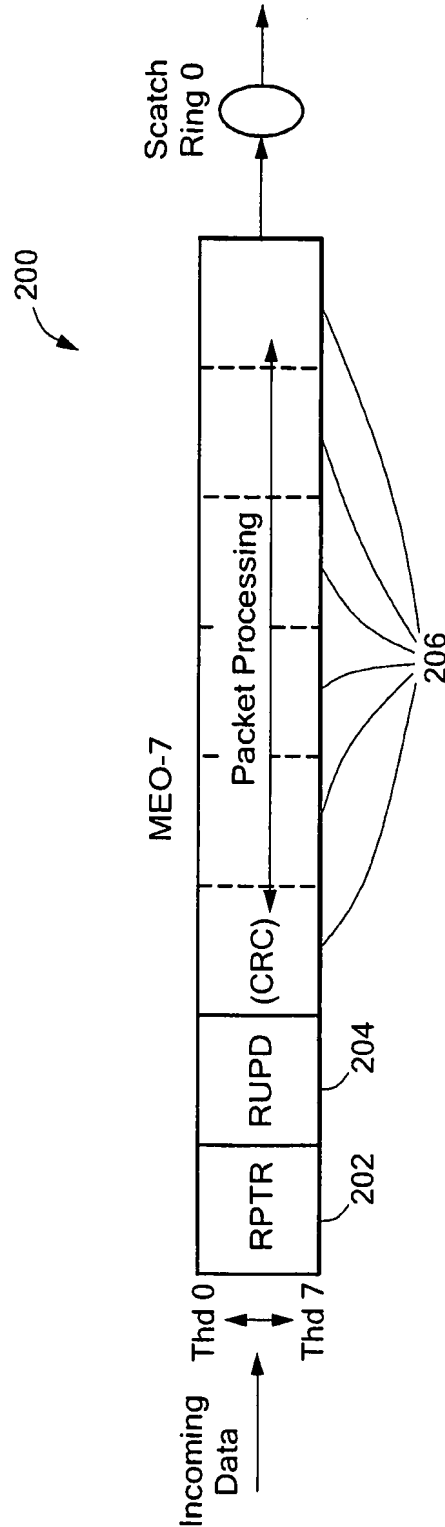


FIG. 13